The Fun Facets of Mario: Multifaceted Experience-Driven PCG via Reinforcement Learning

Ziqi Wang, Jialin Liu
Department of Computer Science and Engineering
Southern University of Science and Technology
Shenzhen, China
12132362@mail.sustech.edu.cn,liujl@sustech.edu.cn

Georgios N. Yannakakis
Institute of Digital Games
University of Malta
Msida, Malta
georgios.yannakakis@um.edu.mt

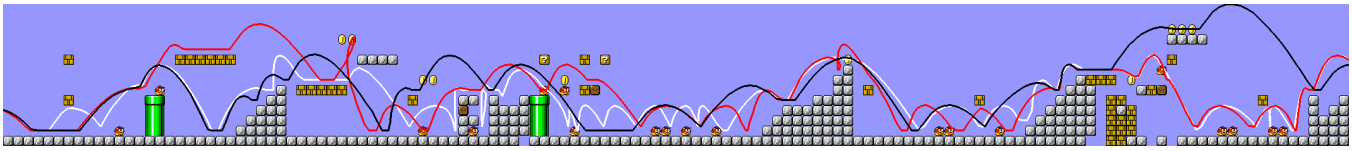


Figure 1: Example of a *Super Mario Bros* level generated by *multifaceted* EDRL. The play traces of three dissimilar agents are also overlaid on the level. EDRL moderates the divergence of game level and gameplay based on Koster's *theory of fun* [9].

ABSTRACT

The recently introduced EDRL framework approaches the experience-driven (ED) procedural generation of game content via a reinforcement learning (RL) perspective. EDRL has so far shown its effectiveness in generating novel platformer game levels endlessly in an online fashion. This paper extends the framework by integrating multiple facets of game creativity in the ED generation process. In particular, we employ EDRL on the creative facets of *game level* and *gameplay* design in *Super Mario Bros.* Inspired by Koster's theory of fun, we formulate *fun* as moderate degrees of level or gameplay divergence and equip the algorithm with such reward functions. Moreover, we enable faster and more efficient game content generation through an episodic generative soft actor-critic algorithm. The resulting *multifaceted* EDRL is not only capable of generating *fun* levels efficiently, but it is also robust with respect to dissimilar playing styles and initial game level conditions.

CCS CONCEPTS

• **Computing methodologies** → *Reinforcement learning*; • **Applied computing** → *Arts and humanities.*

KEYWORDS

Experience-driven procedural content generation, procedural content generation via reinforcement learning, online level generation, platformer games, Super Mario Bros

ACM Reference Format:
Ziqi Wang, Jialin Liu and Georgios N. Yannakakis. 2022. The Fun Facets of Mario: Multifaceted Experience-Driven PCG via Reinforcement Learning. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The autonomous or semi-autonomous generation of games that elicit the *optimal* experience for a player remains a critical challenge for the AI and games research community [41]. More than a decade after the introduction of the experience-driven procedural content generation (EDPCG) framework [40] the idea that content (and not agent behaviour) adapts according to a player's experience has gradually made it to commercial standard games such as *Nevermind* (Flying Mollusk, 2015) and *Spell Forest* (Zynga, 2020). EDPCG was recently blended with other popular frameworks such as the PCG via reinforcement learning (PCGRL) [7] framework. The recently introduced ED(PCG)RL blend, EDRL for short [26], enables the generation of personalised content via the RL paradigm. EDRL was initially tested in *Super Mario Bros* (SMB) (Nintendo, 1985)[17] through the generative designs of RL agents that learn to optimise a number of novelty-based reward functions relevant to level design. That initial study, however, considers only preliminary formulations of player experience for platformer level design.

In this paper, we focus on the notion of *fun* as a player experience state and build formulations of *fun* based on Koster's *theory of fun* [9]. In particular, we train RL agents to *moderate* the level of *divergence* of their own generated content as in [26]. In this subsequent study, however, we build upon and extend the work of Shu *et al.* [26] in a number of ways. First, we introduce the notion of *multifaceted* EDRL generation as we develop experience-driven content creation algorithms across creative facets of games as defined in [11]: *visuals*, *narrative*, *game rules/design*, *audio*, *levels* and *gameplay*. Initially in this study, we consider the two creative facets of *levels* and *gameplay* design and construct ad-hoc formulations of *fun* for both facets (see Fig. 1). We use and quantify the term *fun* in alignment with Koster's definitions; in Section 6 we discuss the limitations of *fun* both as a term and as a quantifiable measure. Second, we perform extensive experiments testing the robustness of EDRL across different reward functions, initial level conditions

and player styles. Finally, the enhanced EDRL framework features an episodic generative soft actor-critic algorithm making it possible to efficiently personalise game generators for individual players.

Prior to running any empirical experiments, the quality of our *fun* formulations is cross-verified by using them to evaluate human designed SMB levels vs randomly generated levels. The *fun* measures that maintain moderate degrees of game level and gameplay variation appear to value human designed levels highly compared to most randomly generated levels. Our EDRL algorithm equipped with such reward functions is able to efficiently generate SMB levels online and it is also highly robust with respect to dissimilar starting level segments and play styles. This study demonstrates the capacity of EDRL and opens up new avenues of research for multifaceted and *orchestrated* [10] generation via EDRL[1].

2 BACKGROUND

In this section we review the literature on player experience modelling in particular, associated with the notion of *fun* (Section 2.1) and then review the ways such measures have been used as heuristics for procedural content generation (Section 2.2).

2.1 Measuring Player Experience

The player experience modelling literature is rich on methods and approaches for modelling aspects of player behaviour and experience [40, 41]. A large body of work has focused on data-driven methods that learn to predict particular player experience states such as frustration, challenge [23, 24] and arousal [16] that are either directly measurable from biosignals [23, 30] or are delivered from player demonstrations of experience [1, 14, 16].

When it comes to the notion of *fun*, most studies in the literature have been inspired by Koster's *theory of fun* [9] suggesting *fun* is the sweet game design spot between game difficulty and player skills. Therefore, quantified notions of *fun* have usually relied on a distance, dissimilarity or diversity measure that considers the level as a whole or as level segments. Shaker *et al.* [25], for instance, machine learned the notion of *fun* through human annotations of various SMB levels. A similar methodology was followed by Martinez *et al.* for simple arcade games [20]. Preuss *et al.*, relied on three types of distance measures, namely *tile-based distance*, *objective-based distance* and *visual impression distance* there were used as functions for evaluating good and diverse game levels [22]. Wuff-Jensen *et al.* used the mean squared error and structure similarity index to evaluate the diversity of digital elevation maps for 3D landscape generation [34]. Variants of the Kullback-Leibler (KL) divergence have been employed to measure the dissimilarity between SMB levels [13], as an objective function for SMB level generation [13], or as a measure of a level's *fun* value [26].

Novelty and *surprise* are notions that are closely linked to dissimilarity and divergence and have been used in association with procedural content generation. In [26] for example, a novelty-based formulation of *fun* and a *historical deviation* metric were used to estimate the quality of generated SMB segments. Gravina *et al.* interpreted *surprise* as the *deviation from expectation*, and introduced an algorithm that maximises the unexpectedness of game content [4]. The notion of surprise has also been applied to quality diversity search [5]. In particular, Gravina *et al.* considered playing behavioural diversity as a component of quality diversity algorithms for content generation [3]. On a similar basis, Khalifa *et al.* generated SMB levels of high quality and with diverse behavioural mechanics using different simulation approaches [8]. Osborn and Mateas developed a tool named *Gamalyzer* to analyse the dissimilarity of play traces based on constraint continuous edit distance [18, 19].

In contrast to all earlier studies, in this paper we employ a number of dissimilarity metrics for measuring the variation of both game levels and player behaviour. To the best of our knowledge, there is no prior attempt available in the literature that explores the multifaceted variation of in-game content and in-game behaviour for content generation. The aim of the study is to build expressive generators that are capable for generating *fun* levels in a multifaceted [11] fashion. Such levels can be tailored to any player via RL, in particular, using the *experience-driven PCG via RL* framework that is reviewed next.

2.2 EDRL Framework

The experience-driven PCG via reinforcement learning [26] framework is built upon the EDPCG [40] and the PCGRL [7] frameworks. EDRL enables online, endless generation of personalised content driven by experience-based reward functions. The framework has demonstrated its efficiency in training SMB level generators that are capable of generating functional and diverse levels considering the level playability and *fun* formulations. Motivated by the effectiveness of EDRL, in this paper we employ EDRL for training SMB level generators using notions of *fun* that maximise the divergence of levels and playing behaviour. We extend earlier work in EDRL [26] substantially in three ways. First, we introduce generic formulations of player experience that are linked to multiple facets of in-game creativity [11]—game level design and gameplay behaviour. Second we propose a training procedure based on soft actor-critic [6] which simulates and rewards complete levels generated by EDRL, thereby yielding better quality (reward) estimates of the level. Finally, we generate and test levels using different player styles.

3 FORMULATING *FUN*?

According to Koster's *theory of fun* [9], a game is *fun* when players learn new skills during gameplay. In terms of player skills and corresponding game challenges, *fun* occurs when the experienced game content is neither too hard (i.e., the player experiences an unmastered game pattern) nor too easy (i.e., the player has already mastered a game pattern). At the same time the game content that is experienced by the player should be expressive enough so that it enables rich gameplay—i.e. diverse ways to experience the content.

The question that arises is what kind of in-game variation affects player experience and needs to be considered by a measure of fun? According to Liapis *et al.* [11], games are multifaceted creative outcomes and they are composed of 6 creative facets: *visuals*, *narrative*, *game rules/design*, *audio*, *levels* and *gameplay*. In this paper we introduce a multifaceted EDRL approach and initially examine the impact of the *game level* and *gameplay* facets of creativity by constructing independent measures of *fun* for each one of them. Those

[1]Code and results available at https://github.com/SUSTechGameAI/MFEDRL

measures are provided to EDRL which, in turn, generates game content that elicits the right amount of variation for each player, thereby maximising *fun*. These two facets are chosen because of their importance to the platformer game genre considered in this study. The level geometry of a platformer game (e.g., position of platforms and placement of enemies) affects directly the player traces available for navigating the level geometry, and vice versa.

In the remainder of this section, we formulate a divergence metric that measures both game level (Section 3.1) and gameplay (Section 3.2) variations, and then propose formulations of *fun* (Section 3.3). In Section 3.4, the fun formulations of the two facets are verified on human designed levels versus randomly generated levels.

3.1 Game Level Divergence

We employ the *tile-pattern divergence* used in [13, 26] to measure the dissimilarity of level segments. In particular, we count the frequency of all distinct tile patterns in a level segment to formulate a tile-pattern frequency distribution, and then compute the divergence between the tile-pattern frequency distributions of a pair of segments. In [13, 26], KL-divergence is used to measure the difference between tile-pattern frequency distributions, in which two hyperparameters w and ϵ are introduced to cater for asymmetry of distributions and non convergence when an uncommon pattern is considered in the calculations. Determining suitable values for w and ϵ is not trivial, however. To tackle this issue, we base our measure of *fun* for game level design on the Jensen Shannon (JS) divergence [12]. $D_L(A, B)$ is a parameter-free measure of JS divergence between any two given level segments A and B and is formulated as follows:

$$D_L(A,B) = \sum_{x \in X(A,B)} \left(p(x|A) \log \frac{p(x|A)}{p(x|A,B)} + p(x|B) \log \frac{p(x|B)}{p(x|A,B)} \right), \quad (1)$$

where $X(A, B)$ is the set of tile-distinct patterns that appear in the segments A and B; $p(x|A)$ denotes the frequency of a tile-pattern x that appears in A; $p(x|A, B) = (p(x|A) + p(x|B))/2$ is the averaged frequency of x appearing in A and B. $D_L(A, B)$ is high when the diversity of x (of a predefined size) is high across two segments A and B. In this work, the size of the tile-pattern is set empirically to 2×2. More details can be found in the code repository[1].

3.2 Gameplay Divergence

Beyond a player's own style, a player's behaviour is affected primarily by the game rules and the game genre. In the platformer game genre examined in this paper, one could consider play traces (i.e., player coordinates over time) as the most representative aspect of player behaviour that is affected directly by the geometry of the level. On that basis, we record a sequence of player positions $\tau(A|\mathcal{P}) = ([x_1, y_1], [x_2, y_2], \cdots, [x_t, y_t], \cdots)$ that represent the behaviour of a player $\mathcal{P}$ on a level segment A, where $[x_t, y_t]$ are the pixel-level coordinates of $\mathcal{P}$'s position at the t-th frame while playing level segment A. To compute the divergence of play traces we employ dynamic time warping (DTW) [2], a method with wide adoption for the analysis of sequential data.

DTW has a clear physical meaning and supports sequences with different lengths as it computes the minimal summation of distance along a *warping path*. Let M and N denote the lengths of $\tau(A|\mathcal{P})$ and $\tau(B|\mathcal{P})$, respectively, and let $K = \max\{M, N\}$. A *warping path* $\mathbf{W} = ([i(k), j(k)] \mid k = 1, 2, \cdots, K)$ is defined as a sequence of paired position indices $[i(k), j(k)]$, where $k \in \{1, 2, \cdots, K\}$, and $i(k) \in \{1, 2, \cdots, M\}$, $j(k) \in \{1, 2, \cdots, N\}$. A warping path satisfies three conditions: (i) it starts from $[1, 1]$ and ends at $[M, N]$; (ii) it never goes backward; and (iii) it covers every entry of the two sequences that are compared. We formulate the DTW dissimilarity of play traces across two level segments A and B for player $\mathcal{P}$ as:

$$D_G(A,B|\mathcal{P}) = \min_{\mathbf{W}} \sum_{k=1}^{K} \mathrm{E}(\tau_{i(k)}(A|\mathcal{P}), \tau_{j(k)}(B|\mathcal{P})). \quad (2)$$

In our case, the distance measurement E in Eq. (2) is the Euclidean distance between two position vectors. To normalise DTW values, D_G in Eq. 2 is divided by the width of segments (i.e. 28 as in [31]).

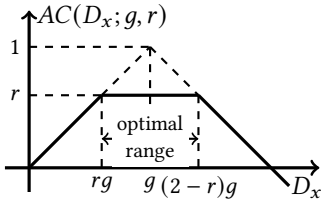


Figure 2: Illustration of the A-clip function used for moderating the variation of game levels and gameplay behavior.

3.3 Moderating Divergence

According to [9] in-game variation should lie within some ideal range. To moderate that amount of variation and thus formulate *fun*, we use the following A-clip function:

$$\mathrm{AC}(D_x; g, r) = \min\{r, 1 - \frac{|D_x - g|}{g}\}. \quad (3)$$

where D_x is either D_L (see Eq. 1) or D_G (see Eq. 2). Parameters g and r ($0 < r \leq 1$) are the target value and the clipping rate of the A-clip function, respectively. The range $[rg, (2-r)g]$ is the optimal range of the A-clip function (see Fig. 2).

In addition to moderating divergence via the A-clip function we also consider the *memory* of the playing experience for quantifying *fun*. Inspired by [26], we assume that the player's memory of experiencing earlier parts of the level decays over time; hence, the importance of moderating divergence should be higher for newly generated level segments compared to level segments played in the past. We therefore let g and n describe, respectively, the target divergence value and the maximum number of previous segments that could be memorised by a player. The degree of *fun*, f_x, of the i-th level segment S_i is formulated as a *slacking A-clipped* function:

$$f_x(S_i; g_x, n) = \frac{\sum_{k=1}^{\min\{n,i\}} \mathrm{AC}(D_x(S_i, S_{i-k}); g_x, r_k)}{\sum_{k=1}^{\min\{n,i\}} r_k}, \quad (4)$$

where $D_x(\cdot, \cdot)$ is the divergence measure of either game level D_L, Eq. (1), or gameplay behaviour D_G, Eq. (2); g_x is a parameter that is set independently for D_L and D_G. $r_k = 1 - \frac{k}{n+1}$ is the rate parameter for moderating divergence between S_i and S_{i-k}. Clearly, the optimal range of moderating divergence between S_i and S_{i-k} keeps slacking as k goes larger. In this paper, we consider Eq. (4) for either game levels (via D_L) or gameplay behaviour (via D_G) as two independent objectives, namely $f_L(\cdot; g_L, n)$, and $f_G(\cdot; g_G, n)$.

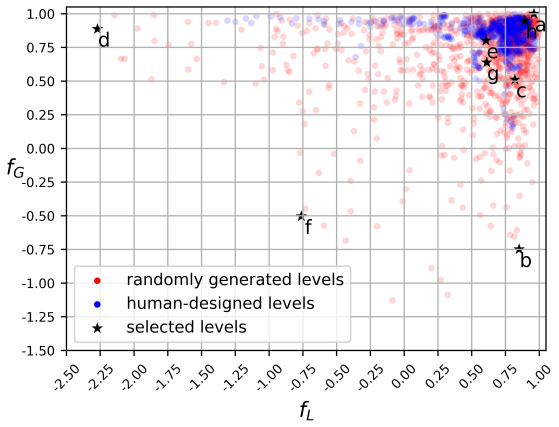


Figure 3: Scatter plot of f values of randomly generated levels (blue) versus human designed levels (red) computed with $g_L = 0.1$, $g_G = 0.25$ and $n = 4$.

3.4 Verifying Fun

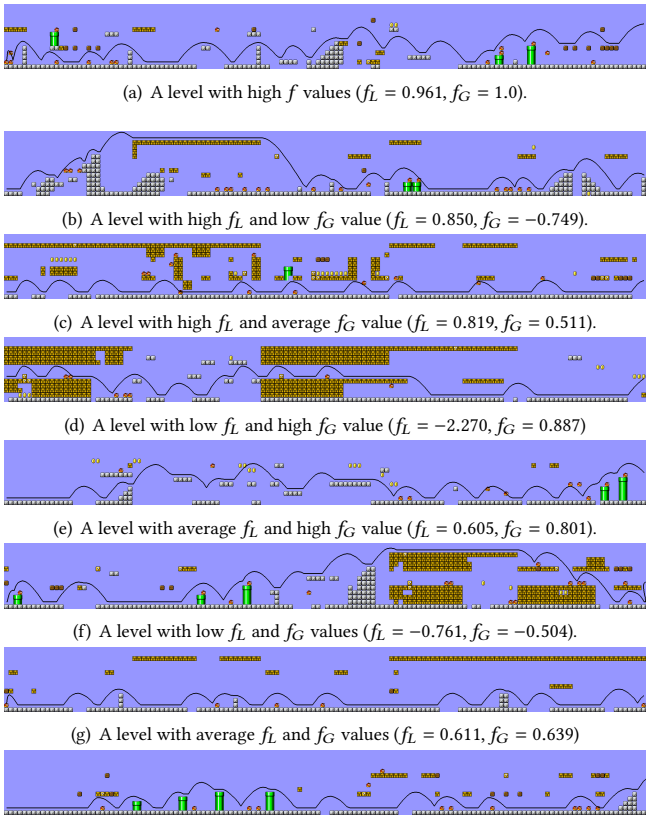
To cross-verify the degree to which our *fun* formulation approximates the experience of *fun*, one would ideally need to solicit labels of *fun* and then correlate the formulation with the solicited human annotations. Such a process was introduced in [36, 37] and followed by a number of studies since then [16, 25, 28]. The alternative approach we examine here is to consider human authored levels of SMB as the ground truth of *fun*. Therefore, we conduct an empirical study contrasting randomly generated and human designed SMB levels assuming that good formulations of *fun* should be able to rate human designed levels highly and also be able to distinguish between *fun* and non-fun (i.e., boring or very challenging) levels among randomly generated ones. Note that a randomly generated level could still be loads of fun.

For that purpose, 1,000 playable levels are generated by concatenating 5 level segments produced by a GAN as in [31] from randomly sampled latent vectors. Each segment is 28-tile width as in [31]. Human designed SMB level pieces are collected by employing a sliding 5-segment width window across all the available levels in the Video Game Level Corpus (VGLC)[2] [29]. In total, we obtain 729 such levels from the human designed SMB levels.

The f_L and f_G scores of each level are computed by averaging the f values across all segments. We perform preliminary sensitivity analysis studies using different hyperparameter values on the randomly generated and human designed levels. The target dissimilarity values of f_L and f_G are set to $g_L = 0.1$ and $g_G = 0.25$, respectively, as those were the values that would distinguish well between *fun* and non-fun levels among randomly generated ones. The maximum number of segments to be considered (n) is set to 4.

Figure 3 illustrates the f scores of randomly generated and human designed levels computed with the selected hyperparameter values. What is obvious from this scatter plot is that human-designed levels—in their majority—are placed in the top right corner of the f_L, f_G map indicating the high correlation between our measures and the quality (i.e., the human engineered player experience)

[2]https://github.com/TheVGLC/TheVGLC



(a) A level with high f values ($f_L = 0.961$, $f_G = 1.0$).

(b) A level with high f_L and low f_G value ($f_L = 0.850$, $f_G = -0.749$).

(c) A level with high f_L and average f_G value ($f_L = 0.819$, $f_G = 0.511$).

(d) A level with low f_L and high f_G value ($f_L = -2.270$, $f_G = 0.887$)

(e) A level with average f_L and high f_G value ($f_L = 0.605$, $f_G = 0.801$).

(f) A level with low f_L and f_G values ($f_L = -0.761$, $f_G = -0.504$).

(g) A level with average f_L and f_G values ($f_L = 0.611$, $f_G = 0.639$)

(h) An indicative human-designed level with high f values ($f_L = 0.894$, $f_G = 0.948$)

Figure 4: Randomly generated levels with dissimilar values of *fun* computed via Eq. (4), and hyperparameters $g_L = 0.1$, $g_G = 0.25$ and $n = 4$ (a–g). A human designed level of high f values is included for comparison (h). The positions of these levels among 1,000 randomly generated levels are indicated with stars and corresponding caption letters in Fig. 3.

of human designed SMB levels. We also observe that most randomly generated levels are scattered in the f_L, f_G map of Fig. 3. Our measures, unsurprisingly, fail to rate all randomly generated levels with high values. To offer more insights on the relationship between a SMB level, its potential playtraces and the resulted f values, some extreme levels are sampled from Fig. 3 and displayed in Fig. 4.

4 EDRL WITH EPISODIC GENERATIVE SAC

In this work, f_L and f_G are considered as two different reward functions, namely R_L and R_G, respectively, that will train SMB level designers using the EDRL framework. Based on the hyperparameter experiments outlined earlier we formulate $R_L(S_i) = f_L(S_i; 0.1, 4)$ and $R_G(S_i) = f_G(S_i; 0.25, 4)$ (cf. Eq. 4). In addition to R_L and R_G, we consider a *playability* reward R_p to make sure the computational designer will generate playable levels. Formally, $R_p(S_i) = -1$ if S_i is playable and $R_p(S_i) = 0$ otherwise.

Both R_G and R_p require agent-based simulations on the generated levels. In traditional RL algorithms, rewards are computed and provided after each time step. Following this traditional approach

implies that levels are simulated on a segment-by-segment basis. Doing so, however, leads to a *truncation bias* as simulating a level segment-by-segment will likely differ from simulating it as a whole. To overcome the challenge of potential truncation biases, we propose an episodic generative soft actor-critic (EGSAC) algorithm, build upon soft actor-critic (SAC) [6], to train level designers via EDRL. SAC is selected as it has proved to outperform many RL algorithms including PPO on a variety of continuous decision making benchmarks and has shown promise in dungeon generation studies [32]. What differentiates EGSAC from the original SAC [6] is that rewards are computed after each episode rather than after each time step. At each episode, EGSAC first generates a complete level using the RL designer, then it repairs and simulates it, and finally it computes rewards for each segment of the level. Once the rewards are computed, the rewarded transitions are added into the replay buffer and the SAC models (actor and critic) are updated with randomly sampled transitions in the replay memory. Algorithm 1 details the EGSAC training procedure that involves five core components, outlined as follows.

GAN generator $\mathcal{G}(\cdot) : \mathbb{Z} \rightarrow \mathbb{S}$. A pre-trained GAN generator is used to map a latent code to a level segment. $\mathbb{Z} = [-1, 1]^{20}$ is the latent vector space and $\mathbb{S}$ is the level segment representation space. Our implementation of GAN is based on *MarioGAN* [31]. To better match the latent vector space of GAN and the action space of the RL designer, latent vectors are sampled uniformly at random rather than using a Gaussian distribution.

RL designer $\pi(\cdot) : \mathbb{Z}^n \rightarrow \mathbb{Z}$. The RL designer is a SAC model composed of an actor and a critic. Two multi-layer perceptrons with 3 hidden layers of 256 neurons each are used for the actor and critic networks, respectively. The RL designer takes the concatenated n most recent latent vectors $[\mathbf{z}_{i-n} : \mathbf{z}_{i-1}]$ as input, and outputs a new latent vector $\mathbf{z}_i$ at each time step i.

Reward function $\mathcal{R}(\cdot, \cdot) : \mathbb{S}^{m+n} \times \mathbb{T}^{m+n} \rightarrow \mathbb{R}^m$. $\mathbb{T}$ and $\mathbb{R}$ are the play trace space and real number space, respectively. m is the length of episode. The reward function is computed from the i^{th} to the $(m+n)^{th}$ segments and their corresponding play traces, while the first n segments are needed to compute f_L and f_G.

Repairer $\mathcal{F}(\cdot) : \mathbb{S}^{m+n} \rightarrow \mathbb{S}^{m+n}$. A repairer is used to repair broken pipes in the levels generated by GAN. The algorithm is based on the repairer introduced in [27]; its efficiency, however, is further improved by using a *divide-and-conquer* strategy: a level is split into a number of pieces with the same number of pipe tiles. Then, the pieces are repaired separately and those pieces are then concatenated to yield a complete repaired level.

Simulated player $\mathcal{P}$. An agent is used to simulate a player throughout the complete level. If the simulation terminates before the agent reaches the rightmost point of the level, the agent will be reset to 1-tile away from its terminal position. In this paper, we use three agents with different play styles (playing personas), namely *runner*, *killer* and *collector*. The *runner* is an A* agent, which aims at completing the level as soon as possible. The *killer* and *collector* agents introduced in [42] aim, respectively, at killing more enemies and collecting more coins while completing the game level. All three are based on the A* algorithm but employing different cost and heuristic functions. Section 5.3 provides more details of those three agents when discussing experimental results.

Algorithm 1 EGSAC. In all experiments, $m = 25$, $T = 10$, $B = 384$, $p = 50$. Hyperparameter n of Eq. (4) is 4.

Require: Length of episode m; Interval of updating the networks T; Batch size of updating the networks B; number of processes p; GAN generator $\mathcal{G}(\cdot)$; RL designer $\pi(\cdot)$; Reward function $\mathcal{R}(\cdot, \cdot)$; Repairer $\mathcal{F}(\cdot)$; Simulated player $\mathcal{P}$

Initialise an empty replay memory $\mathcal{D}$
repeat ▷ *Lines 3-12 run in parallel across p processes*
  for $i = 0 \rightarrow n - 1$ **do**
    Randomly initialise $\mathbf{z}_i \sim U(-1, 1)$
    $S_i \leftarrow \mathcal{G}(\mathbf{z}_i)$
  end for
  for $i = n \rightarrow n + m - 1$ **do**
    $\mathbf{s}_i \leftarrow [\mathbf{z}_{i-n} : \mathbf{z}_{i-1}]$, $\mathbf{z}_i \leftarrow \pi(\mathbf{s}_i)$, $S_i \leftarrow \mathcal{G}(\mathbf{z}_i)$
  end for
  $[S_0 : S_{n+m-1}] \leftarrow \mathcal{F}([S_0 : S_{n+m-1}])$
  $[\tau_0 : \tau_{n+m-1}] \leftarrow$ Simulate $\mathcal{P}$ on $[S_0 : S_{n+m-1}]$
  $[r_n : r_{n+m-1}] \leftarrow \mathcal{R}([S_0 : S_{n+m-1}], [\tau_0 : \tau_{n+m-1}])$
  for $i = n \rightarrow n + m - 1$ **do**
    Store $\langle \mathbf{s}_i, \mathbf{z}_i, r_i \rangle$ into $\mathcal{D}$
  end for
  for $u = 1 \rightarrow \lfloor m * \frac{p}{T} \rfloor$ **do**
    $\mathcal{B} \leftarrow$ Uniformly sample a mini-batch of size B from $\mathcal{D}$
    Update the SAC model with $\mathcal{B}$
  end for
until run out of training budget

The use of our EGSAC implementation for EDRL has two core benefits: it reduces the truncation bias and it improves the training efficiency. The episodic computation of rewards enables repairing and simulating a complete level rather than a segment. On that basis, the divide-and-conquer strategy we employ to the repairer was both deemed to be necessary and yields faster training of EDRL designers. Experimental tests show that repairing and simulating 25 successive segments via EGSAC is about 5 times faster than repairing and simulating every individual segment as in earlier work [26]: 10.54 and 49.96 seconds, respectively, averaged over 100 trials. Moreover, our implementation of EGSAC supports repairing and simulating multiple levels using parallel processes.

5 EXPERIMENTS

This section presents experiments with EDRL across different reward functions, different initial conditions and different player types. Designers are trained with the same procedure on the same machine with an Intel Xeon Gold 6240 CPU and a RTX 2080Ti GPU.

5.1 Varying the Reward Function

In the first set of our experiments EDRL designers are trained for $1e6$ time steps (segments). EDRL uses different reward functions that consider *fun* attributed to the level design (R_L) and the gameplay behaviour (R_G) independently or together. In all experiments reported here, we employ the *runner* agent to train the EDRL designers.

Table 1 summarises the key results across 100 independent trials. It becomes clear that all EDRL trained designers are capable of generating playable levels. The designer trained with the sum of

Table 1: Mean reward and standard deviation values across segments. Values are averaged across 100 independent trials of 25 generated segments each. The best and worst results are highlighted in bold and italics, respectively. A random designer (bottom row) is also added as a baseline. All of our trained designers are capable of generating playable levels.

Reward	R_L	R_G	R_p
$R_G + R_p$	0.685 ± 0.06	0.938 ± 0.04	-0.005 ± 0.01
$R_L + R_p$	0.974 ± 0.03	0.826 ± 0.08	**-0.000 ± 0.00**
$R_L + R_G + R_p$	**0.975 ± 0.03**	**0.946 ± 0.03**	-0.001 ± 0.01
Random	*0.493 ± 0.25*	*0.536 ± 0.20*	*-0.172 ± 0.07*

R_L and R_G performs better than the other two designers tested on both R_L and R_G. All the designers outperform the random designer significantly. The EGSAC algorithm employed in this paper is efficient as all training scenarios examined converge quickly. More experimental results can be found in the code repository[1].

As an indicative example of EDRL generation in Fig. 5 we visualise the 10 segments generated by the 3 different EDRL designers as starting from level (g) of Fig. 4; in these examples we illustrate the play trace of an A* agent. As seen in Fig. 5 (a), the EDRL designer trained solely to design levels that yield diverse enough gameplay behaviours (via R_G) generates levels that look far less diverse. Even though the play traces are varied enough, the game level patterns appear to be similar, giving the level a monotonous feel. On the other hand, the EDRL designer trained solely with R_L (see Fig. 5 (b)) generates level segments that alternate patterns so that content variation is kept within appropriate bounds. The EDRL designer trained to maximise both game level and gameplay rewards (R_L+R_G; Fig. 5 (c)) seems to generate more interesting and varied levels with regards to both level and behavioural patterns.

5.2 Varying the Initial Conditions

The robustness of our EDRL designers is evaluated as follows: we first select the latent vectors of 3 outlier random initialisations indicated by levels (b), (d) and (g) of Fig. 4, we then employ our 3 trained designers on those initial observations, and request them to generate 10 new segments. We record the mean values of R_L and R_G—using a sliding window of size 4 and stride of 1—across the 10 generated segments. In Fig. 6 we plot the f_L and f_G values as they evolve across the 10 generated segments. Our EDRL designers appear to be robust with respect to the initial segment they are presented with as all of them manage to reach the top right corner of Fig. 6. Interestingly, the EDRL designer trained with R_L initially heads towards the lower right corner—i.e., increase of f_L but decrease of f_G values—but then manages to recover and increase both f values. Qualitatively this behaviour suggests that it is necessary to first improve the *fun* values for gameplay behaviour prior to improving *fun* associated with the game level. A similar phenomenon also appears with the EDRL designer trained with the R_G only.

5.3 Varying the Player

In the last round of experiments we fix the reward function to $R_G + R_L + R_p$ and use different agents as simulated players. We employ the 3 different agents described earlier: the *runner*, the *killer* and the *collector*. We evaluate those designers with all of the agents and present the obtained results in Table 2. We can observe that designers trained with the corresponding player yield, unsurprisingly, higher values for R_G and R_p than for other player personas; the only counter example is the *playability* reward of the *runner* agent. One possible reason for this anomaly lies in the *runner's* ability to complete levels faster than the other two agents; thus, training designers using other agents makes them generate easier levels for the *runner* agent. The agent used during training also influences the corresponding R_L score positively. It appears, however, that the EDRL designer trained with the *killer* player finds it more challenging to moderate the divergence of level design (R_L=0.920) compared to the other two agents (R_L = 0.974 for the runner and R_L = 0.956 for the collector agent).

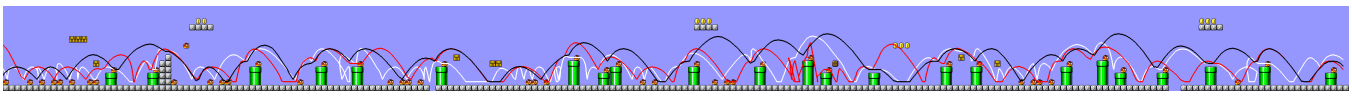
Figure 7 visualises the levels generated by the different EDRL designers using the same initial conditions (i.e. initial segments). It is evident that the resulted levels are very different from each another. Interestingly, it seems that the designer trained with the *killer* persona generates fewer monster enemies, while the designer trained with the *collector* persona generates fewer coins, compared to the EDRL designer trained with the *runner* agent. Such a finding verifies the principle of *moderated divergence* quantified by our f formulation. It further indicates that generating way too many coins and enemies yields way too divergent player behaviours, respectively, for the *killer* and the *collector* persona and, in turn, results to levels with lower f values.

6 DISCUSSION

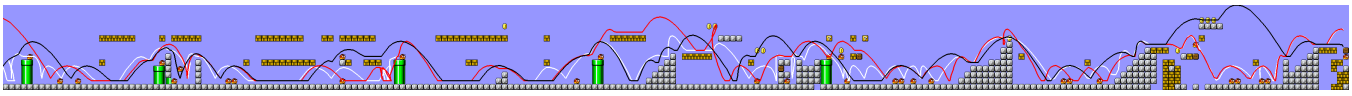
In this paper we introduced formulations of fun based on Koster's *theory of fun* and used those as reward functions for the EDRL generative framework. We observed the efficiency and robustness of the introduced EGSAC algorithm to generate endless SMB levels that are playable and *fun* across the creative facets of game level design and gameplay [11]. Given that this paper introduced EDRL for experience-driven multifaceted generation, in this section, we discuss a number of limitations and research avenues that need to be explored in future studies.

Although EDRL appears to be robust to initial conditions, the generated levels highly depend on the reward function and the behaviour of the test agent. In addition to the 3 agents tested in our experiments, more human-like agents and reward functions will need to be studied in future work to further encourage the RL agent to learn to select suitable level segments for different types of players. Using data and play traces (demonstrations) from human-like agents, our EDRL may continually (online) learn through such behaviour and adapt to the player's skill and preferences.

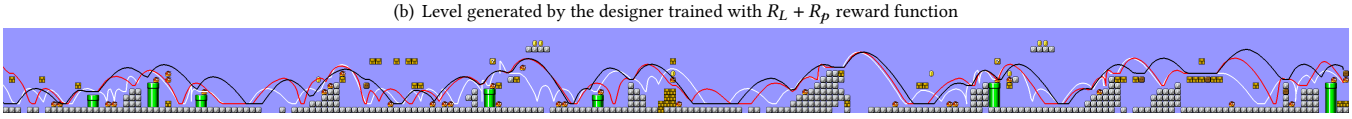
In the proposed implementation of EDRL, we used a number of metrics to directly represent player experience (in particular *fun*) in a theory-driven manner [41]. In addition to *fun*, we intend to generate levels with adaptive levels of diversity, surprise [5, 39] or novelty as in [26]. Importantly, we wish to extend the fun formulations presented here to the 4 remaining facets—where relevant and appropriate—including visuals, narrative, audio and game rules [11]. Similarly to [3, 21], we aim to apply multi-objective optimisation to consider simultaneously the various facets of *fun* in game



(a) Level generated by the designer trained with $R_G + R_p$ reward function



(b) Level generated by the designer trained with $R_L + R_p$ reward function



(c) Level generated by the designer trained with $R_L + R_G + R_p$ reward function, using *runner* agent for both training and testing.

Figure 5: EDRL designers seeded with level (g) of Fig. 4 as the initial segment and trained with different rewards. Black, white and red curves illustrate the play traces of the *runner*, *killer* and *collector* agents, respectively.

Table 2: Mean reward values (and standard deviation values) across segments. Values are averaged across 100 independent trials of 25 generated segments each. Columns represent evaluation metrics and conditions (reward|agent) whereas rows represent which agent is used during the training of the EDRL designer. The best and worst results are highlighted in bold and italics, respectively. Cells with underlined values correspond to the agents that are used during training.

Training Player	R_G | *Runner*	R_G | *Killer*	R_G | *Collector*	R_p | *Runner*	R_p | *Killer*	R_p | *Collector*	R_L
Runner	**0.945 ± 0.03**	*0.465 ± 0.13*	*0.621 ± 0.13*	-0.002 ± 0.01	*-0.682 ± 0.09*	*-0.467 ± 0.12*	**0.974 ± 0.03**
Killer	*0.774 ± 0.05*	**0.878 ± 0.06**	0.873 ± 0.07	**-0.001 ± 0.01**	**-0.040 ± 0.04**	-0.047 ± 0.05	*0.920 ± 0.03*
Collector	0.870 ± 0.06	0.860 ± 0.10	**0.900 ± 0.06**	*-0.007 ± 0.02*	-0.184 ± 0.14	**-0.046 ± 0.05**	0.956 ± 0.02

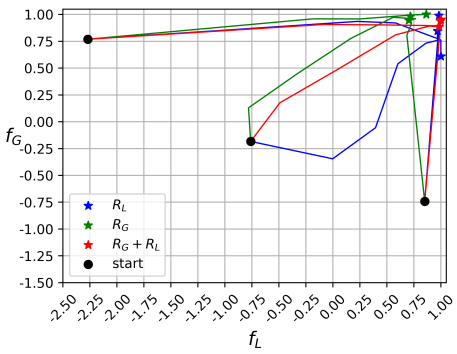


Figure 6: The *fun* path (f_L and f_G value pairs) of EDRL generators starting from levels (b), (d) and (g) of Fig. 4.

(a) Level generated by the designer trained with the *runner* player

(b) Level generated by the designer trained with the *killer* player

(c) Level generated by the designer trained with the *collector* player

Figure 7: Generating levels with different simulated players given an identical initial segment. Black, white and red curves refer to play traces of the *runner*, *killer* and *collector*, respectively.

generation, instead of their linear aggregation with fixed weights. The current fun-based reward formulations and any ad-hoc designed reward functions based on player experience will need to be cross-verified and tested against human players as in [37] or against publicly available SMB annotated levels [28]. Our findings from this paper, however, suggest that the two ad-hoc designed measures of *fun* manage to value human-designed levels higher than most random levels. Nevertheless, our assumptions are strong as we hypothesise that the ground truth of player experience (defined by the term fun) can be found in human-designed levels. Arguably not all players enjoy all levels out there which calls for user studies.

There is often a lot of critique and discussion about the limitations of *fun* both as a term and any attempt to quantify it. Earlier studies have defined a long list of arguments against [35] but also for the use of the term in applied AI research in games [38]. Arguably the term is not easily measured, it is not supported by any theory of emotion, it is not measured by any sensor available, yet people can still express it, game design frameworks like Koster's defined it, and thus one can even attempt to measure it via, e.g., affective computing methods [15]. One additional limitation is that the definition of *fun* used in this paper appears to be objectively

defined as a function of game levels and behaviour diversity. EDRL, however, is trained on three different play styles and hence *fun* is subjective to these playing styles and can be personalised to a player's archetype. To our best knowledge, there is no universal way to define and formulate fun comprehensively. Our planned user studies will consider the variation of human play for the personalised generation of *fun* content for different play styles.

While EDRL is only tested on platformer games, we expect it to operate for any game generation task. Our plan is to test EDRL on more complex games that feature large game and action space representations. Visualising the latent space and corresponding levels [33] is also worth exploring in follow up studies.

7 CONCLUSION

In this paper we introduce an experience-driven generative paradigm [40] via reinforcement learning [7] that is able to design *fun* facets of game creativity [11]. In particular, we extend the EDRL framework [26] and train RL agents to design endless and playable levels that maximise notions of *fun* [9] for game *level* and *gameplay* design. We employ a pre-trained GAN generator that designs level segments for Super Mario Bros considering moderate degrees of game level and gameplay diversity. Interestingly, the *fun* formulation introduced in this paper appears to value human-designed SMB levels higher than most randomly generated ones. The efficiency of the EDRL framework is improved via an episodic generative soft actor-critic method for rapid training of SMB generators. Our key findings suggest that EDRL is not only effective and fast in generating *fun* levels for players, it is also robust with regards to different initial conditions and player types.

REFERENCES

[1] Matthew Barthet, Antonios Liapis, and Georgios N Yannakakis. 2021. Go-Blend behavior and affect. In *2021 9th International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW)*. IEEE, 1–8.

[2] Donald J Berndt and James Clifford. 1994. Using dynamic time warping to find patterns in time series.. In *KDD workshop*, Vol. 10. Seattle, WA, USA:, 359–370.

[3] Daniele Gravina, Ahmed Khalifa, Antonios Liapis, Julian Togelius, and Georgios N Yannakakis. 2019. Procedural content generation through quality diversity. In *2019 IEEE Conference on Games*. IEEE, 1–8.

[4] Daniele Gravina, Antonios Liapis, and Georgios Yannakakis. 2016. Surprise Search: Beyond Objectives and Novelty. In *Proc. of GECCO 2016*. ACM, 677–684.

[5] Daniele Gravina, Antonios Liapis, and Georgios N Yannakakis. 2018. Quality diversity through surprise. *IEEE Trans. on Evolutionary Computation* 23, 4 (2018), 603–616.

[6] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of International Conference on Machine Learning*. PMLR, 1861–1870.

[7] Ahmed Khalifa, Philip Bontrager, Sam Earle, and Julian Togelius. 2020. PCGRL: Procedural content generation via reinforcement learning. *Proceedings of the Sixteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 16, 1 (2020), 95–101.

[8] Ahmed Khalifa, Michael Cerny Green, Gabriella Barros, and Julian Togelius. 2019. Intentional computational level design. In *Proceedings of The Genetic and Evolutionary Computation Conference*. 796–803.

[9] Raph Koster. 2013. *Theory of fun for game design.* " O'Reilly Media, Inc.".

[10] Antonios Liapis, Georgios N Yannakakis, Mark J Nelson, Mike Preuss, and Rafael Bidarra. 2018. Orchestrating game generation. *IEEE Trans. on Games* 11, 1 (2018), 48–68.

[11] Antonios Liapis, Georgios N Yannakakis, and Julian Togelius. 2014. Computational game creativity. ICCC.

[12] Jianhua Lin. 1991. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information theory* 37, 1 (1991), 145–151.

[13] Simon M. Lucas and Vanessa Volz. 2019. Tile pattern KL-divergence for analysing and evolving game levels. In *Proceedings of the Genetic and Evolutionary Computation Conference*. Association for Computing Machinery, 170–178.

[14] Konstantinos Makantasis, Antonios Liapis, and Georgios N Yannakakis. 2021. The pixels and sounds of emotion: General-purpose representations of arousal in games. *IEEE Transactions on Affective Computing* (2021).

[15] Héctor P Martínez and Georgios N Yannakakis. 2014. Deep multimodal fusion: Combining discrete events and continuous signals. In *Proceedings of the 16th International conference on multimodal interaction*. 34–41.

[16] David Melhart, Antonios Liapis, and Georgios N Yannakakis. 2022. The Arousal video Game AnnotatIoN (AGAIN) Dataset. *IEEE Trans. on Affective Computing* (2022).

[17] R&D4 Nintendo. 1985. Super Mario Bros. *Game [NES].(13 September 1985). Nintendo, Kyoto, Japan* (1985).

[18] Joseph C Osborn and Michael Mateas. 2014. A game-independent play trace dissimilarity metric. In *Proc. of FDG*.

[19] Joseph Carter Osborn, Ben Samuel, Joshua Allen McCoy, and Michael Mateas. 2014. Evaluating play trace (dis) similarity metrics. In *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference*.

[20] Héctor Perez Martínez, Maurizio Garbarino, and Georgios N Yannakakis. 2011. Generic physiological features as predictors of player experience. In *International Conference on Affective Computing and Intelligent Interaction*. Springer, 267–276.

[21] Thomas Pierrot, Valentin Macé, Felix Chalumeau, Arthur Flajolet, Geoffrey Cideron, Karim Beguir, Antoine Cully, Olivier Sigaud, and Nicolas Perrin-Gilbert. 2022. Diversity Policy Gradient for Sample Efficient Quality-Diversity Optimization. In *Proceedings of GECCO*. ACM.

[22] Mike Preuss, Antonios Liapis, and Julian Togelius. 2014. Searching for good and diverse game levels. In *Proc. of IEEE CIG*. 1–8.

[23] Noor Shaker, Stylianos Asteriadis, Georgios N Yannakakis, and Kostas Karpouzis. 2013. Fusing visual and behavioral cues for modeling user experience in games. *IEEE transactions on cybernetics* 43, 6 (2013), 1519–1531.

[24] Noor Shaker, Julian Togelius, and Mark J Nelson. 2016. *Procedural content generation in games*. Springer.

[25] Noor Shaker, Georgios N. Yannakakis, and Julian Togelius. 2010. Towards automatic personalized content generation for platform games. In *Proceedings of AIIDE*, Vol. 6. 63–68.

[26] Tianye Shu, Jialin Liu, and Georgios N. Yannakakis. 2021. Experience-driven PCG via reinforcement learning: A Super Mario Bros study. In *2021 IEEE Conference on Games*. 1–9. https://doi.org/10.1109/CoG52621.2021.9619124

[27] Tianye Shu, Ziqi Wang, Jialin Liu, and Xin Yao. 2020. A novel CNet-assisted evolutionary level repairer and its applications to Super Mario Bros. In *2020 IEEE Congress on Evolutionary Computation*. IEEE, 1–10.

[28] Kristin Siu, Matthew Guzdial, and Mark O Riedl. 2017. Evaluating singleplayer and multiplayer in human computation games. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*. 1–10.

[29] Adam James Summerville, Sam Snodgrass, Michael Mateas, and Santiago Onta n'on Villar. 2016. The VGLC: The Video Game Level Corpus. *Proceedings of the 7th Workshop on Procedural Content Generation* (2016).

[30] Simone Tognetti, Maurizio Garbarino, Andrea Bonarini, and Matteo Matteucci. 2010. Modeling enjoyment preference from physiological responses in a car racing game. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*. IEEE, 321–328.

[31] Vanessa Volz, Jacob Schrum, Jialin Liu, Simon M. Lucas, Adam Smith, and Sebastian Risi. 2018. Evolving Mario levels in the latent space of a deep convolutional generative adversarial network. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, Association for Computing Machinery, 221–228.

[32] Mariana Werneck and Esteban WG Clua. 2020. Generating procedural dungeons using machine learning methods. In *2020 19th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. IEEE, 90–96.

[33] Oliver Withington and Laurissa Tokarchuk. 2022. Compressing and Comparing the Generative Spaces of Procedural Content Generators. In *2022 IEEE CoG*.

[34] Andreas Wulff-Jensen, Niclas Nerup Rant, Tobias Nordvig Møller, and Jonas Aksel Billeskov. 2017. Deep convolutional generative adversarial network for procedural 3D landscape generation based on DEM. In *Interactivity, Game Creation, Design, Learning, and Innovation*. Springer, 85–94.

[35] Georgios N Yannakakis. 2008. How to model and augment player satisfaction: a review. ICMI.

[36] Georgios N Yannakakis and John Hallam. 2005. A generic approach for generating interesting interactive pac-man opponents. (2005).

[37] Georgios N Yannakakis and John Hallam. 2007. Towards optimizing entertainment in computer games. *Applied Artificial Intelligence* 21, 10 (2007), 933–971.

[38] Georgios N Yannakakis, John Hallam, and Henrik Hautop Lund. 2008. Entertainment capture through heart rate activity in physical interactive playgrounds. *User Modeling and User-Adapted Interaction* 18, 1 (2008), 207–243.

[39] Georgios N Yannakakis and Antonios Liapis. 2016. Searching for surprise. ICCC.

[40] Georgios N. Yannakakis and Julian Togelius. 2011. Experience-driven procedural content generation. *IEEE Trans. on Affective Computing* 2, 3 (2011), 147–161.

[41] Georgios N. Yannakakis and Julian Togelius. 2018. *Artificial Intelligence and Games*. Springer.

[42] Keyuan Zhang, Jiayu Bai, and Jialin Liu. 2022. Generating Game Levels of Diverse Behaviour Engagement. In *2022 IEEE Conference on Games (CoG)*.